Docker and PHP

Running PHP Applications in Containers

Review: Why Docker for PHP?

The PHP Development Problem:

```
👨 Student A: "My PHP 8.2 code works!"
👩 Student B: "I have PHP 8.1, getting errors..."
👨 Student C: "I'm on Windows, can't install extensions..."
👦 Professor: "Let's use Docker instead!"
```

Docker Solution: Same PHP environment for everyone

PHP Without Docker vs With Docker

Traditional PHP Setup:

```
# Different for each OS
sudo apt install php8.2            # Ubuntu
brew install php@8.2               # Mac
# Download XAMPP installer         # Windows

# Install extensions
sudo apt install php8.2-mysql php8.2-json php8.2-mbstring...
# Configure php.ini...
# Set up web server...
# 😫 1-2 hours of setup time
```

Docker PHP Setup:

Make sure Docker Desktop or Docker daemon is running.

```
# Same command for all OS
docker run -it php:8.2-cli php --version
```

Result: PHP 8.2 running in 30 seconds on any computer!

No need to install/configure PHP locally — Docker handles it.

🐳 The Magic Behind Docker Run

1. Image Pull

- php:8.2-cli is an official PHP Docker image (based on Linux).
- If not present locally, Docker downloads it automatically.

2. Container Start

- docker run -it → start a new container
- Runs PHP in an isolated Linux environment

3. Command Execution

- Docker runs: `php --version`
- Shows PHP version directly inside the container

Basic PHP Container Example

Let's Run PHP in a Container:

```
# Run PHP interactively in the Docker container
docker run -it php:8.2-cli bash

# Inside container:
php --version
# PHP 8.2.12 (cli) (built: Oct 26 2023 13:37:39) (NTS)

# Create and run a PHP file
echo "<?php echo 'Hello from Docker' . PHP_EOL;" > hello.php
php hello.php
# Hello from Docker!
```

Magic: PHP 8.2 works perfectly, no installation needed!

Running Your PHP Code with Docker

Problem: How to run MY PHP files?

Create a simple PHP script on your host machine:

```php
<?php
// test.php
echo "Hello, I'm PHP " . PHP_VERSION . "\n";
echo "Current time: " . date('Y-m-d H:i:s') . "\n";
?>
```

How do we run this with Docker?

Solution: Volume Mapping

```
# Map current directory to container
docker run --rm -v $(pwd):/app -w /app php:8.2-cli php test.php
```

Breakdown:

- `--rm`: Remove container after it finishes (you do not need `--rm`, but it is generally good practice to use it for single-run commands to avoid container clutter)
- `-v $(pwd):/app`: Mount current directory to `/app` in container
- `-w /app`: Set working directory to `/app`
- `php:8.2-cli`: Use PHP 8.2 command-line image
- `php test.php`: Run our PHP script

Understanding the Command

```
docker run --rm -v $(pwd):/app -w /app php:8.2-cli php test.php
```

What happens:

It assumes that you run the command in the `/home/user` directory.

```
Your Computer                Docker Container
┌──────────────────┐        ┌──────────────────┐
│  /home/user/     │        │     /app/        │
│   ├── test.php   │ ←────→ │    ├── test.php  │
│   └── ...        │        │    └── ...       │
└──────────────────┘        │                  │
                            │  PHP 8.2 Engine  │
                            └──────────────────┘
```

Key Insight: Container can access your files, but runs in an isolated environment

Practical Example: Simple PHP API

Create a REST API endpoint:

```php
<?php
// api.php
header('Content-Type: application/json');

$method = $_SERVER['REQUEST_METHOD'];

if ($method === 'GET') {
    $response = [
        'message' => 'Hello from Docker PHP!',
        'php_version' => PHP_VERSION,
        'timestamp' => time(),
        'server' => 'Docker Container'
    ];
    echo json_encode($response, JSON_PRETTY_PRINT);
} else {
    http_response_code(405);
    echo json_encode(['error' => 'Method not allowed']);
}
```

Run the API with Built-in Server:

```
# Start the PHP built-in web server in Docker
docker run --rm -p 8000:8000 -v $(pwd):/app -w /app \
  php:8.2-cli php -S 0.0.0.0:8000
```

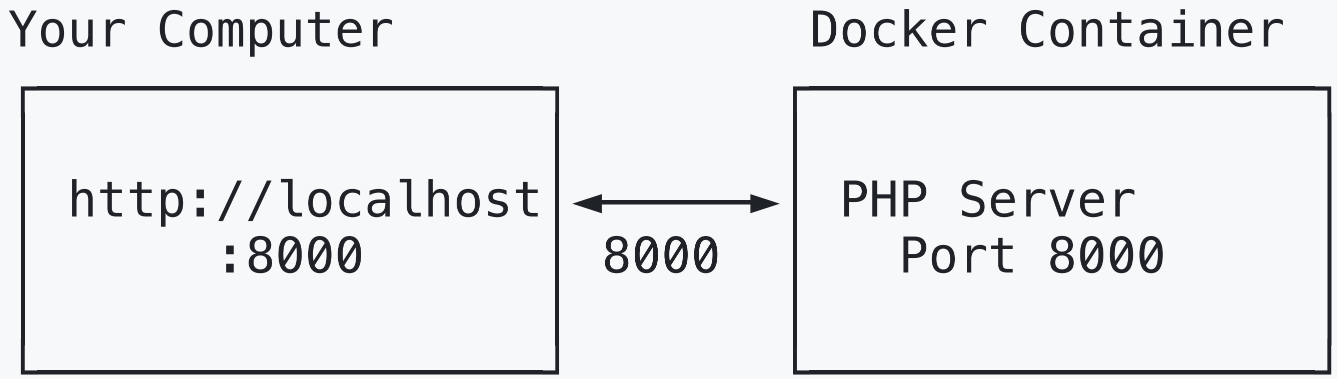
New parts:

- `-p 8000:8000` : Map port 8000 from container to host
- `php -S 0.0.0.0:8000` : Start PHP built-in web server

Test it: Open `http://localhost:8000/api.php` in your browser!

```
{
    "message": "Hello from Docker PHP!",
    "php_version": "8.2.29",
    "timestamp": 1757690632,
    "server": "Docker Container"
}
```

Understanding Port Mapping



Port Mapping: `-p host_port:container_port`

Creating a Dockerfile for PHP

Problem: Typing long commands is tedious

```
# This is getting long...
docker run --rm -p 8000:8000 -v $(pwd):/app -w /app \
  php:8.2-cli php -S 0.0.0.0:8000
```

Solution: Create a Dockerfile

```dockerfile
# Dockerfile
FROM php:8.2-cli

# Set working directory
WORKDIR /app

# Copy PHP files
COPY . .

# Expose port 8000
EXPOSE 8000

# Start the PHP built-in server
CMD ["php", "-S", "0.0.0.0:8000"]
```

Using Your Custom Dockerfile

Build your image:

```
# Build image with tag "my-php-app"
docker build -t my-php-app .
```

Run your container:

```
# Much simpler!
docker run --rm -p 8000:8000 my-php-app
```

Benefits:

- ✅ Shorter command
- ✅ Reproducible environment
- ✅ Easy to share with others

Dockerfile Explained Line by Line

```dockerfile
FROM php:8.2-cli
# Start with official PHP 8.2 command-line image

WORKDIR /app
# Set /app as the working directory inside the container

COPY . .
# Copy all files from the current directory to /app in the container

EXPOSE 8000
# Document that this container listens on port 8000

CMD ["php", "-S", "0.0.0.0:8000"]
# Default command when container starts
```

Is the Dockerfile Required for Docker?

`No` in this case (You can use Dockerfile though):

- If the official/prebuilt image already provides:
  - The runtime (e.g., `nginx:alpine`, `mysql:8`, `redis:7`)
  - And you only configure via **volumes** or **environment variables**
- Example: NGINX server with your config & HTML
  - ✅ Use `image: nginx:alpine`
  - ❌ No Dockerfile needed

Use image: only if the base image works as-is.

`Yes` in this case:

- If you must **extend or customize** the base image:
  - Add **libraries or packages**
  - Install **PHP extensions** (`pdo_mysql` , `gd` , etc.)
  - Include **Composer**, `php.ini` , or scripts
  - Bundle **app code** directly into the image

```
FROM php:8.2-fpm
RUN docker-php-ext-install pdo_mysql
COPY . /var/www/html
```

Use build: + Dockerfile when you need extra functionality.

Development Workflow with Docker

1. Write Your PHP Code

```php
<?php
// index.php
echo "Welcome to my PHP app!\n";
echo "PHP Version: " . PHP_VERSION . "\n";
?>
```

2. Create Dockerfile

```dockerfile
FROM php:8.2-cli
WORKDIR /app
COPY . .
EXPOSE 8000
CMD ["php", "-S", "0.0.0.0:8000"]
```

3. Build and Run

```
# Build once
docker build -t my-app .

# Run for development (with volume for live editing)
docker run --rm -p 8000:8000 -v $(pwd):/app my-app

# Or run production version (code baked into image)
docker run --rm -p 8000:8000 my-app
```

4. Test Your App

Visit `http://localhost:8000` in your browser